NASA-TM-111964

A FAST METHOD FOR THE NUMERICAL EVALUATION OF CONTINUOUS FOURIER AND LAPLACE TRANSFORMS*

DAVID H. BAILEY[†] AND PAUL N. SWARZTRAUBER[‡]

Abstract. The fast Fourier transform (FFT) is often used to compute numerical approximations to continuous Fourier and Laplace transforms. However, a straightforward application of the FFT to these problems often requires a large FFT to be performed, even though most of the input data to this FFT may be zero and only a small fraction of the output data may be of interest. In this note, the "fractional Fourier transform," previously developed by the authors, is applied to this problem with a substantial savings in computation.

Key words. fast Fourier transform, fractional Fourier transform, numerical integration

AMS subject classifications. 65T20. 65T10. 65Y20

1. Introduction. The continuous Fourier transform (CFT) of a function $f(t)$ (which may have real or complex values) and its inverse will be defined here as

$$ (1) \qquad F[f](x) = \int_{-\infty}^{\infty} f(t)e^{-itx}dt, $$

$$ (2) \qquad F^{-1}[f](x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(t)e^{itx}dt, $$

where i is the imaginary unit. The discrete Fourier transform (DFT) and the inverse DFT of an n-long sequence z (which may have real or complex values) will be defined here as

$$ (3) \qquad D_k(z) = \sum_{j=0}^{n-1} z_j e^{-2\pi ijk/m}, $$

$$ (4) \qquad D_k^{-1}(z) = \frac{1}{m} \sum_{j=0}^{n-1} z_j e^{2\pi ijk/m}. $$

Straightforward evaluation of the DFT using these formulas is expensive, even when the exponential factors are precomputed. This cost can be greatly reduced by employing one of the variants of the fast Fourier transform (FFT) algorithm [1], [2], [5], [6].

The methods developed in this paper are equally applicable to the numerical evaluation of continuous Laplace transforms, and the approach is the same as for Fourier transforms. For brevity, this exposition will focus on evaluating continuous Fourier transforms.

2. A conventional scheme for evaluating CFTs with FFTs. There are of course a number of advanced techniques for the approximate numerical evaluation of definite integrals [4]. Unfortunately, most of these techniques deal with evaluating only one integral, rather than a large number of integrals, each with a different integrand. Evaluating the CFT falls in this latter category, since one usually requires the values of this integral for a large range of x.

The FFT can be effectively applied to this problem as follows. Let us assume that $f(t)$ is zero outside the interval $(-a/2, a/2)$. Let $\beta = a/m$ be the interval in t for the m input values of $f(t)$, which are assumed centered at zero, where m is even. To be specific, the abscissas for the input data are $t_j = (j - m/2)\beta$, $0 \leq j < m$. The abscissas for the output data set will

be defined as $x_k = 2\pi(k - m/2)/a = 2\pi(k - m/2)/(m\beta)$, $0 \leq k < m$. This definition of x_k will be explained later. Then we can write

$$(5) \qquad F(x_k) = \int_{-\infty}^{\infty} f(t)e^{-itx_k}dt$$

$$(6) \qquad = \int_{-a/2}^{a/2} f(t)e^{-itx_k}dt$$

$$(7) \qquad \approx \sum_{j=0}^{m-1} f(t_j)e^{-it_j x_k}\beta$$

$$(8) \qquad = \beta \sum_{j=0}^{m-1} f(t_j)e^{-2\pi i(j-m/2)(k-m/2)/m}$$

$$(9) \qquad = \beta e^{\pi i(k-m/2)} \sum_{j=0}^{m-1} f(t_j)e^{\pi ij}e^{-2\pi ijk/m}$$

$$(10) \qquad = (-1)^k \beta D_k[\{(-1)^j f(t_j)\}], \qquad 0 \leq k < m.$$

The DFT indicated in (10) can of course be rapidly evaluated using an FFT.

It is now clear why x_k was defined as above—this is necessary for the expression (9) to be in the form of a DFT. In other words, the interval (i.e., resolution) of the output results of this procedure is fixed at the value $2\pi/(m\beta)$ as soon as one specifies the number m of input values and their interval β.

Let us assume that comparable intervals are required in t and x to obtain accurate results. Then one must have $\beta \approx 2\pi/(m\beta)$, or in other words $m \approx 2\pi/\beta^2$. From this observation it is clear that if one wishes to obtain accurate, high-resolution results using this procedure, then it may be necessary to set m very large, perhaps much larger than the actual size of the nonzero input data set. Another way of saying this is that the input data $f(t_j)$ may need to be padded on both sides with many zeros to obtain the desired resolution in the output data.

A specific example will illustrate these issues. Let us consider the problem of numerically computing the Fourier transform of the Gaussian probability density function

$$(11) \qquad f(t) = \frac{1}{\sqrt{2\pi}}e^{-t^2/2}.$$

It is well known that the Fourier transform of $f(t)$ is

$$(12) \qquad F(x) = e^{-x^2/2}.$$

Note that outside the interval $[-10, 10]$, we have $f(t) < 7.7 \times 10^{-23}$ and $F(x) < 1.9 \times 10^{-22}$.

Suppose now that one wishes to compute the Fourier transform using the above FFT scheme with a resolution of $\sqrt{2\pi}/256$ in both t and x. One can do this provided that one sets $m = 65,536$, since in that case $\beta = 2\pi/(m\beta)$. However, both the input and output data then span the interval $[-m\beta/2, m\beta/2]$, or approximately $[-320.8, 320.8]$.

In short, as a consequence of the fundamentally inflexible nature of the FFT when applied to this problem, it turns out that a large fraction of the input data to the FFT used to evaluate (10) are considerably smaller than the machine "epsilon" of most computers, even when using double precision arithmetic. Also, only a small fraction (the central values) of the output of this FFT are of interest. Thus we seek a more flexible scheme for problems of this sort, one that can produce the central section of the results, using only the central nonzero input data, with a savings of computation.

3. The Fractional Fourier Transform (FRFT). We will employ here a generalization of the DFT that has been termed the FRFT [3]. It is defined on the m-long complex sequence x as

$$(13) \qquad G_k(x, \alpha) = \sum_{j=0}^{m-1} x_j e^{-2\pi i j k \alpha}.$$

The parameter α is not restricted to rational numbers and in fact may be any complex number. Note that the ordinary DFT and its inverse are special cases of the FRFT.

A fortunate fact is that the FRFT admits computation by means of a fast algorithm, one with complexity comparable to the FFT. In fact, it employs the FFT in a crucial step. This algorithm can be stated as follows. Define the $2m$-long sequences y and z as

$$(14) \qquad y_j = x_j e^{-\pi i j^2 \alpha}, \qquad 0 \le j < m,$$

$$(15) \qquad y_j = 0, \qquad m \le j < 2m,$$

$$(16) \qquad z_j = e^{\pi i j^2 \alpha}, \qquad 0 \le j < m,$$

$$(17) \qquad z_j = e^{\pi i (j-2m)^2 \alpha}, \qquad m \le j < 2m.$$

It is then shown in [3] that

$$(18) \qquad G_k(x, \alpha) = e^{-\pi i k^2 \alpha} D_k^{-1}[\{D_j(y) D_j(z)\}], \qquad 0 \le k < m.$$

The remaining m results of the final inverse DFT are discarded. Element-wise complex multiplication is implied in the expression $D_j(y) D_j(z)$. These three DFTs can of course be efficiently computed using $2m$-point FFTs.

To compute a different m-long segment $G_{k+s}(x, \alpha)$, $0 \le k < m$, it is necessary to slightly modify the above procedure. In this case z is as follows:

$$(19) \qquad z_j = e^{\pi i (j+s)^2 \alpha}, \qquad 0 \le j < m,$$

$$(20) \qquad z_j = e^{\pi i (j+s-2m)^2 \alpha}, \qquad m \le j < 2m.$$

Then we have

$$(21) \qquad G_{k+s} = e^{-\pi i (k+s)^2 \alpha} D_k^{-1}[\{D_j(y) D_j(z)\}], \qquad 0 \le k < m.$$

Note that the exponential factors in (14)–(21) can be precomputed. Furthermore, the DFT of the z sequence can also be precomputed. Thus the cost of an m-point FRFT is only about four times the cost of an m-point FFT.

4. Computing CFTs with FRFTs. One of the applications of FRFTs mentioned in [3] is computing DFTs of sparse sequences, i.e., sequences that are mostly zero. It is particularly effective when only a small portion of the input data values are nonzero, and only a small portion of the output values are required. Thus, the FRFT is very well suited for the problem at hand.

As before, we will assume that $f(t)$ is zero outside the interval $(-a/2, a/2)$, and that $\beta = a/m$ is the interval of the m input values of $f(t)$, which will be assumed centered at zero; i.e., $t_j = (j - m/2)\beta$, $0 \le j < m$. We will now define γ to be the interval desired for the output data, so that they are given by $x_k = (k - m/2)\gamma$, $0 \le k < m$. Set $\delta = \beta \gamma / (2\pi)$. Then we can write

$$(22) \qquad F(x_k) = \int_{-\infty}^{\infty} f(t) e^{-i t x_k} dt$$

$$(23) \qquad = \int_{-a/2}^{a/2} f(t)e^{-itx_k}dt$$

$$(24) \qquad \approx \sum_{j=0}^{m-1} f(t_j)e^{-it_j x_k}\beta$$

$$(25) \qquad = \beta \sum_{j=0}^{m-1} f(t_j)e^{-i(j-m/2)(k-m/2)\beta\gamma}$$

$$(26) \qquad = \beta e^{\pi i(k-m/2)m\delta} \sum_{j=0}^{m-1} f(t_j)e^{\pi ijm\delta}e^{-2\pi ijk\delta}$$

$$(27) \qquad = \beta e^{\pi i(k-m/2)m\delta} G_k[\{f(t_j)e^{\pi ijm\delta}\}, \delta], \qquad 0 \le k < m.$$

We now have an economical means of computing the central m results of this transform. Additional m-long segments of results can be obtained by applying the more general form of the FRFT given at the end of the previous section.

5. Implementation example. Collecting the results we have obtained so far, we have

$$(28) \qquad F(x_k) = \int_{-\infty}^{\infty} f(t)e^{-itx_k}dt$$

$$(29) \qquad \approx \beta e^{\pi i(k-m/2)m\delta} G_k[\{f(t_j)e^{\pi ijm\delta}\}, \delta]$$

$$(30) \qquad = \beta e^{\pi i[(k-m/2)m\delta-k^2\delta]} D_k^{-1}[\{D_j(y)D_j(z)\}], \qquad 0 \le k < m,$$

where the $2m$-long sequences y and z are given by

$$(31) \qquad y_j = f(t_j)e^{\pi i(jm\delta-j^2\delta)}, \qquad 0 \le j < m,$$

$$(32) \qquad y_j = 0, \qquad m \le j < 2m,$$

$$(33) \qquad z_j = e^{\pi ij^2\delta}, \qquad 0 \le j < m,$$

$$(34) \qquad z_j = e^{\pi i(j-2m)^2\delta}, \qquad m \le j < 2m.$$

Let us now reconsider the problem of numerically computing the Fourier transform of the Gaussian probability density function

$$(35) \qquad f(t) = \frac{1}{\sqrt{2\pi}}e^{-t^2/2}.$$

Recall that the Fourier transform of $f(t)$ is

$$(36) \qquad F(x) = e^{-x^2/2}.$$

Also recall that outside the interval $[-10, 10]$, we have $f(t) < 7.7 \times 10^{-23}$ and $F(x) < 1.9 \times 10^{-22}$.

To compute the Fourier transform using the conventional FFT scheme with a resolution of $\sqrt{2\pi}/256$ in both t and x, we have $\beta = \gamma = \sqrt{2\pi}/256$ and $m = 2\pi/(\beta\gamma) = 65,536$. An implementation of this scheme requires 3.95 seconds on a SGI Personal Iris. Compared with exact values of the transform computed from the formula above, the root-mean-square (RMS) error of the central 2048 results is 1.14×10^{-16}.

Note however that only the central 2048 result values, which span the approximate interval $[-10.02, 10.02]$, are required—outside this interval both the input and output function values

can be assumed to be zero with high accuracy. Applying the FRFT scheme with $m = 2048$ yields these same results in 0.40 seconds on the Iris, fully ten times faster. The RMS error of the FRFT results is 2.96×10^{-16}. Both of these error figures are virtually at the limit of machine precision.

As an aside, while it is expected from the results of this paper that the FRFT results should be within machine precision of the FFT results, it is remarkable that both are also within machine precision of the actual values resulting from the analytic formula (36). This is in spite of the fact that the interval used in the step function approximation to the integral has the modestly small value $\beta \approx 0.01$. This surprising phenomenon evidently derives from the fact that step function approximations to integrals converge to the exact results very rapidly provided that the function $f(t)$ has rapidly decreasing Fourier coefficients [4]. This condition is met by the Gaussian probability density function.

It should also be mentioned that in this specific example, one may accurately recover the original data by performing an inverse transform with the FRFT. This can be done by employing formulas (30)–(34) with δ replaced by $-\delta$. This operation works because the results of the forward transform, like the original data, can be assumed to be precisely zero outside the interval $[-10.02, 10.02]$. For other functions $f(t)$ this condition may not hold, and as a result this inversion operation would not be expected to accurately reconstruct the original data.

6. More advanced quadrature schemes. The techniques described above were formulated on the basis of a simple step-function approximation to the integral. These techniques can also be applied to a Simpson's rule approximation, as follows. Let $c_j = 4$ for odd j and $c_j = 2$ for nonzero even j. Then we have

$$(37) \qquad F(x_k) = \int_{-\infty}^{\infty} f(t)e^{-itx_k} dt$$

$$(38) \qquad = \int_{-a/2}^{a/2} f(t)e^{-itx_k} dt$$

$$(39) \qquad \approx \sum_{j=0}^{m-1} c_j f(t_j)e^{-it_j x_k} \beta$$

$$(40) \qquad = \beta \sum_{j=0}^{m-1} c_j f(t_j)e^{-i(j-m/2)(k-m/2)\beta\gamma}$$

$$(41) \qquad = \beta e^{\pi i(k-m/2)m\delta} \sum_{j=0}^{m-1} c_j f(t_j)e^{\pi ijm\delta} e^{-2\pi ijk\delta}$$

$$(42) \qquad = \beta e^{\pi i(k-m/2)m\delta} G_k[\{c_j f(t_j)e^{\pi ijm\delta}\}, \delta], \qquad 0 \le k < m.$$

One difficulty with using Simpson's rule to evaluate Fourier integrals is that the accuracy of the quadrature depends on the size of the product $f(t)e^{-itx}$. In particular, the accuracy may deteriorate for large values of x, due to the oscillatory nature of the integrand. For this reason some scientists prefer using Filon's method [4, p. 151] to evaluate Fourier integrals. Filon's method has the advantage that the accuracy depends only on the smoothness of $f(t)$. Again, an FRFT-based scheme can be profitably applied here, since Filon's method employs equispaced intervals. The approach is entirely similar to the above, and details will not be given here.

In general, the FRFT-based technique described above can be profitably applied to the numerical evaluation of any integral transform where (1) both the input function values $f(t_j)$

and the output transform values $F(x_k)$ are equally spaced, (2) a large fraction (more than 75%) of $f(t_j)$ are either zero or within machine tolerance of zero relative to max $|f(t_j)|$, and (3) only a limited range of $F(x_k)$ is required.

REFERENCES

[1] D. H. BAILEY, *FFTs in external or hierarchical memory*, J. Supercomputing, 4 (1990), pp. 23–35.

[2] ———, *A high-performance FFT algorithm for vector supercomputers*, International J. Supercomputer Appl., 2 (1988), pp. 82–87.

[3] D. H. BAILEY AND P. N. SWARZTRAUBER, *The fractional Fourier transform and applications*, SIAM Rev., 33 (1991), pp. 389–404.

[4] P. J. DAVIS AND P. RABINOWITZ, *Methods of Numerical Integration*, Wiley, New York, 1984.

[5] P. N. SWARZTRAUBER, *Multiprocessor FFTs*, Parallel Comput., 5 (1987), pp. 197–210.

[6] C. VAN LOAN, *Computational Frameworks for the Fast Fourier Transform*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.